AMENDMENTS

This section presents changes to the specification and the claims in a clean-unmarked format. A version with markings to show the , changes made by the current amendment is provided after the remarks section.

In The Abstract:

Please replace the abstract beginning at page 40, line 2 with the paragraph:

A1

-- A method and apparatus for executing partial-width packed data instructions are discussed. The processor may include a plurality of registers, a register renaming unit, a decoder, and a partial-width execution unit. The register renaming unit provides an architectural register file to store packed data operands each of which include a plurality of data elements. The decoder is to decode a first and second set of instructions that each specify one or more registers in the architectural register file. The first set of instructions specify operations to be performed on all of the data elements stored in the one or more specified registers. In contrast, the second set of instructions specify operations to be performed on only a subset of the data elements. The partial-width execution unit is to execute operations specified by either the first or the second set of instructions. --

In The Specification:

Page 1, line 2 insert the following paragraph:

A7

-- This application is a continuation patent application of Application Serial No. 09/053,127, entitled *Executing Partial-Width Packed Data Instructions*, and filed on March 31, 1998, NOW U.S. PAT. NO. 6,230,253. --

A

Please replace the paragraph beginning at page 9, line 9 with the paragraph:

A2

-- Another situation in which it is undesirable to force a packed data instruction to return individual results for each pair of data elements includes arithmetic operations in an environment providing partial-width hardware. Due to cost and/or die limitations, it is common not to provide full support for certain arithmetic operations, such as divide. By its nature, the divide operation is very long, even when full-width hardware (e.g., a one-to-one correspondence between execution units and data elements) is implemented. Therefore, in an environment that supports only full-width packed data operations while providing partial-width hardware, the latency becomes even longer. As will be illustrated further below, a partial-width packed data operation, such as a partial-width packed data divide operation, may selectively allow certain portions of its operands to bypass the divide hardware. In this manner, no performance penalty is incurred by operating upon only a subset of the data elements in the packed data operands. --

Please replace the paragraph beginning at page 10, line 1 with the paragraph:

A3

-- Additionally, exceptions raised in connection with extraneous data elements may cause confusion to the developer and/or incompatibility between SISD and SIMD machines. Therefore, it is advantageous to report exceptions for only those data elements upon which the instruction is meant to operate. Partial-width packed data instruction support allows a predictable exception model to be achieved by limiting the triggering of exceptional conditions to those raised in connection with the data elements being operated upon, or in which exceptions produced by extraneous data elements would be likely to cause confusion or incompatibility between SISD and SIMD machines. --

A

Please replace the paragraph beginning at page 33, line 9 with the paragraph:

A4

-- As described above, packed data operations may be implemented as two half width micro instructions (e.g., a high order operation and a low order operation). Rather than independently decoding the macro instruction by two decoders to produce the high and low order operations as would be typically required by prior processor implementations, as a feature of the present embodiment both micro instructions may be generated by the same decoder. In this example, this is accomplished by replication logic 1150 which replicates either the high or low order operation and subsequently modifies the resulting replicated operation appropriately to create the remaining operation. Importantly, as was described earlier, by carefully encoding the register address space, the registers referenced by the micro instructions (e.g., the logical source and destination registers) can be made to differ by a single bit. As a result, the modification logic 1160 in its most simple form may comprise one or more inverters to invert the appropriate bits to produce a high order operation from a low order operation and vice versa. In any event, the replicated micro instruction is then passed to multiplexer 1170. The multiplexer 1170 also receives a micro instruction produced by decoder 1120. In this example, the multiplexer 1170, under the control of a validity decoder 1180, outputs the replicated micro instruction for packed data operations (including partial-width packed data operations) and outputs the micro instruction received from decoder 1120 for operations other than packed data operations. Therefore, it is advantageous to optimize the opcode map to simplify the detection of packed data operations by the replication logic 1150. For example, if only a small portion of the macro instruction needs to be examined to distinguish packed data

A

A4

operations from other operations, then less circuitry may be employed by the validity decoder 1180. --

Please replace the paragraph beginning at page 35, line 13 with the paragraph:

A5

--In an implementation that passes through source data elements to the logical destination register for purposes of executing partial-width packed data operations, in addition to selection logic similar to that described with respect to **Figures 5A and 5C**, logic may be included to eliminate ("kill") one of the high or low order operations. Preferably, for performance reasons, the extraneous micro instruction is eliminated early in the pipeline. This elimination may be accomplished according to the embodiment depicted by using a micro instruction selection signal output from micro instruction length determination circuitry 1190. The micro instruction length determination logic 1190 examines a portion of the macro instruction and produces the micro instruction selection signal which indicates a particular combination of one or more micro instructions that are to proceed down the pipeline. In the case of a scalar SIMD instruction, only one of the resulting high and low order operations will be allowed to proceed. For example, the micro instruction selection signal may be represented as a bit mask that identifies those of the micro instructions that are to be retained and those that are to be eliminated. Alternatively, the micro instruction selection signal may simply indicate the number of micro instructions from a predetermined starting point that are to be eliminated or retained. Logic required to perform the elimination described above will vary depending upon the steering mechanism that guides the micro instructions through the remainder of the pipeline. For instance, if the micro instructions are queued, logic may be added to manipulate the head and tail pointers of the micro instruction queue to cause invalid micro instructions to be overwritten by subsequently generated valid micro instructions.

A